# Tarea Evaluable Docker - 2EV_Ejercicio_1_Contenedores en red y Docker Desktop

Módulo: Despliegue de aplicaciones Web - Distancia

Alumno: Alfonso Dapena Cores



### 1.1 Consideraciones generales

- ¿Qué es Docker?

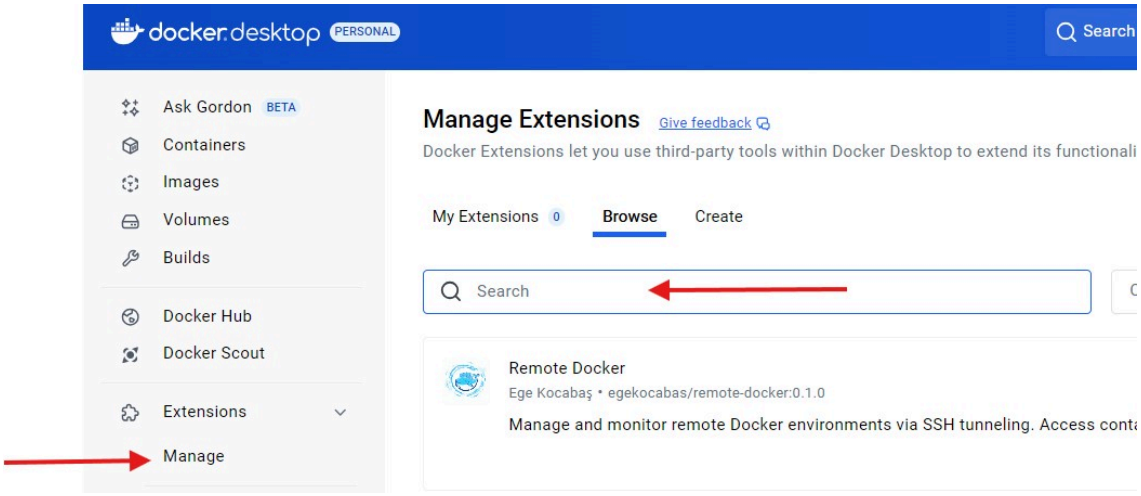
  Docker es una plataforma de código abierto que permite crear, ejecutar y gestionar aplicaciones en contenedores. Los contenedores son unidades estandarizadas que combinan el código de la aplicación con sus dependencias. De manera similar a cómo una máquina virtual virtualiza el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.

  Con Docker, podemos utilizar los contenedores como máquinas virtuales livianas y modulares, y obtener la flexibilidad necesaria para crearlos, implementarlos, copiarlos y trasladarlos de un entorno a otro. Docker permite empaquetar y ejecutar una aplicación en un entorno ligeramente aislado llamado contenedor. El aislamiento y la seguridad permiten ejecutar varios contenedores simultáneamente en un host. Los contenedores son ligeros y contienen todo lo necesario para ejecutar la aplicación, por lo que no es necesario depender de lo que esté instalado en el host. Podemos compartir contenedores mientras trabajamos y nos aseguramos de que todos los usuarios con los que lo compartimos obtengan el mismo contenedor, que funciona de la misma manera.

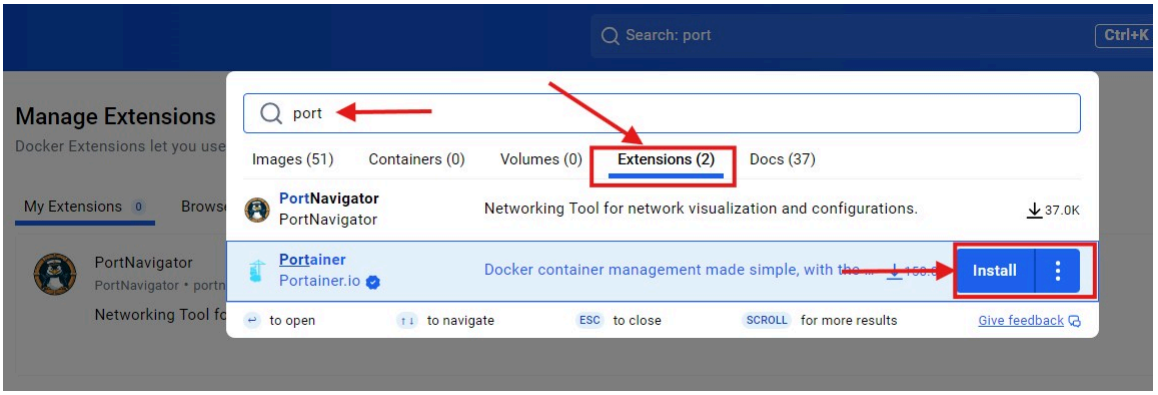
  Docker proporciona herramientas y una plataforma para administrar el ciclo de vida de sus contenedores, denominada Docker Hub, que es un repositorio público de imágenes de contenedores que permite almacenar, administrar y compartir imágenes de Docker, que son las plantillas que definen un contenedor, es decir, el espacio para ejecutar aplicaciones y servicios. Las imágenes Docker contienen el código, las bibliotecas y dependencias necesarias para que nuestro contenedor funcione.

### 1.2 Creación red bridge `redge1`

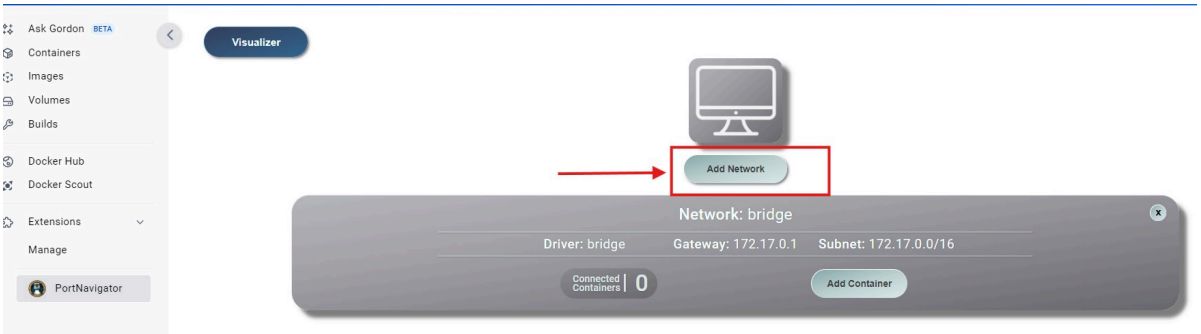
- En primer lugar, accedemos a la aplicación Docker Desktop, pinchamos en la pestaña `Extensions` y desplegamos la opción `Manage`:



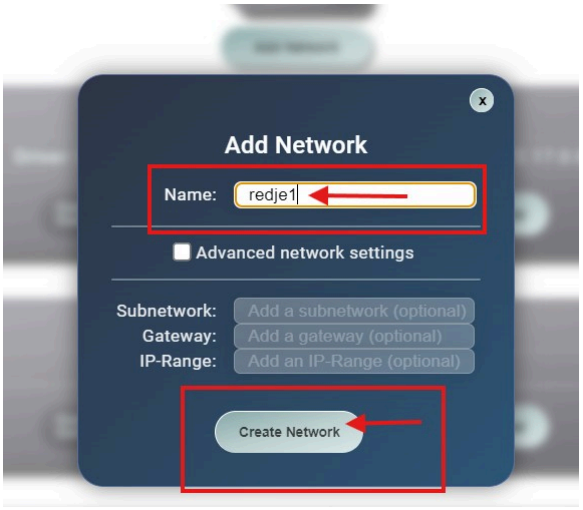

- Buscamos la extensión `PortNavigator` y la descargamos:

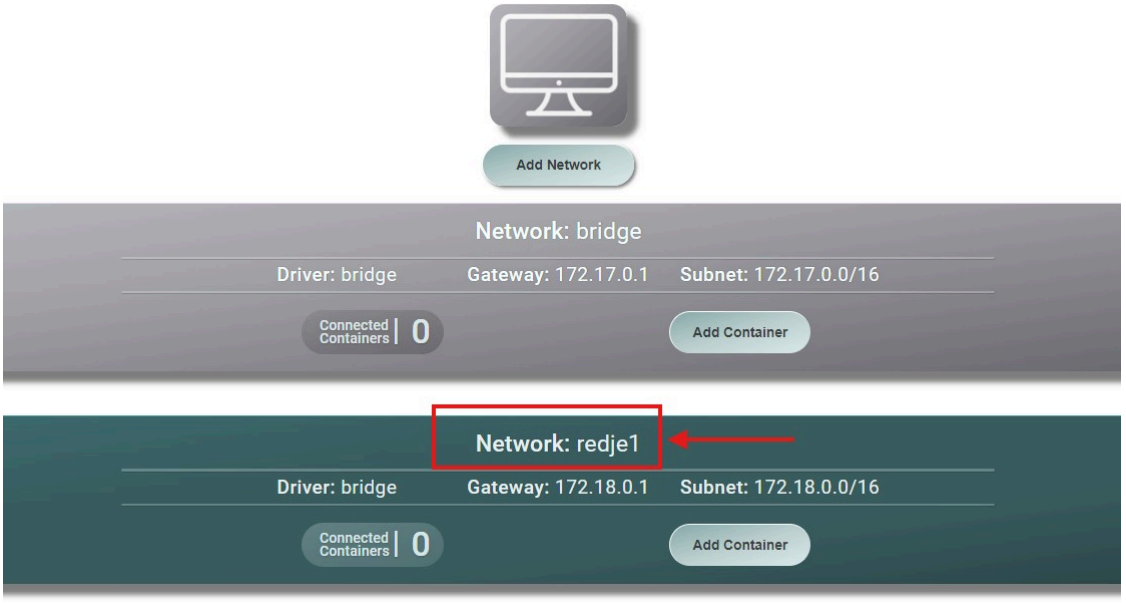



- A continuación, crearemos la red brigde `redje1` para acceder a la pestaña de la extensión `PortNavigator`, y pinchamos en la opción `Add Network`:



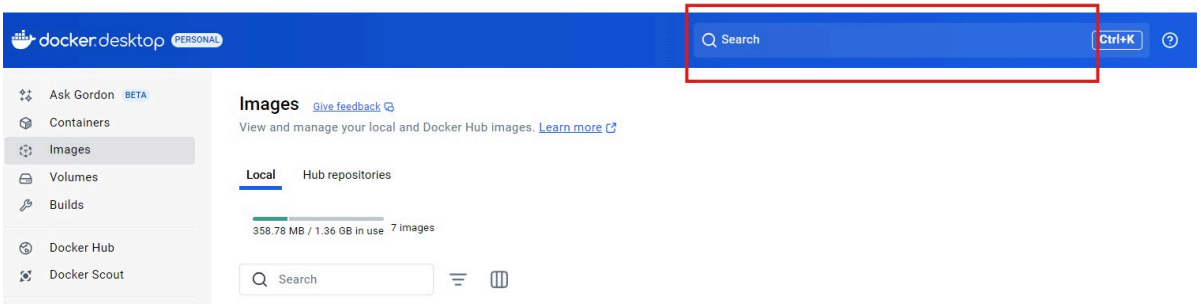

- Le damos el nombre a la red y la creamos. Por defecto el propio programa asignará las direcciones IP a nuestra red.



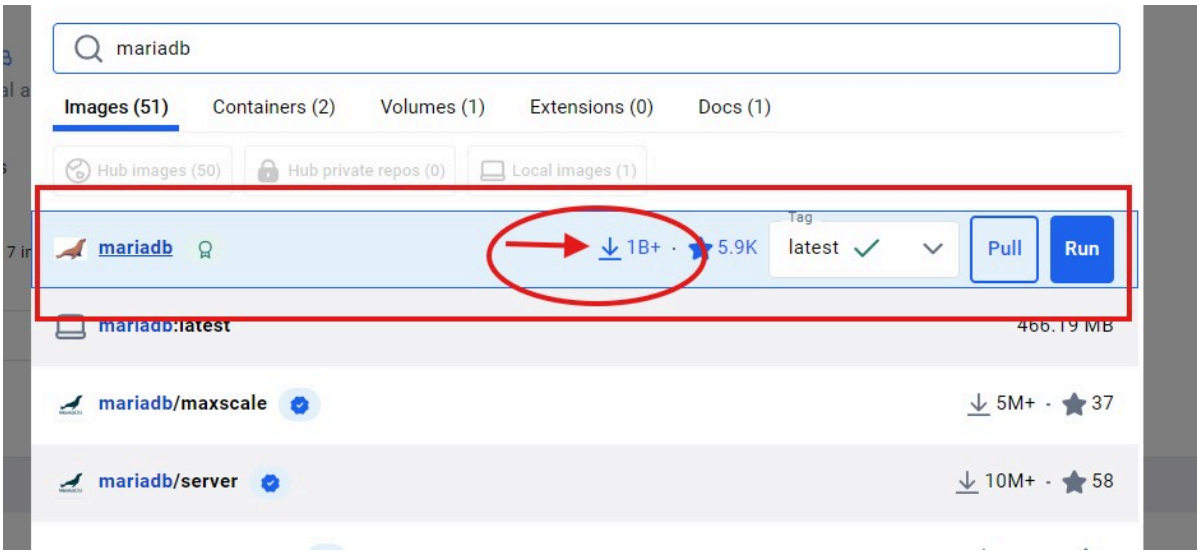



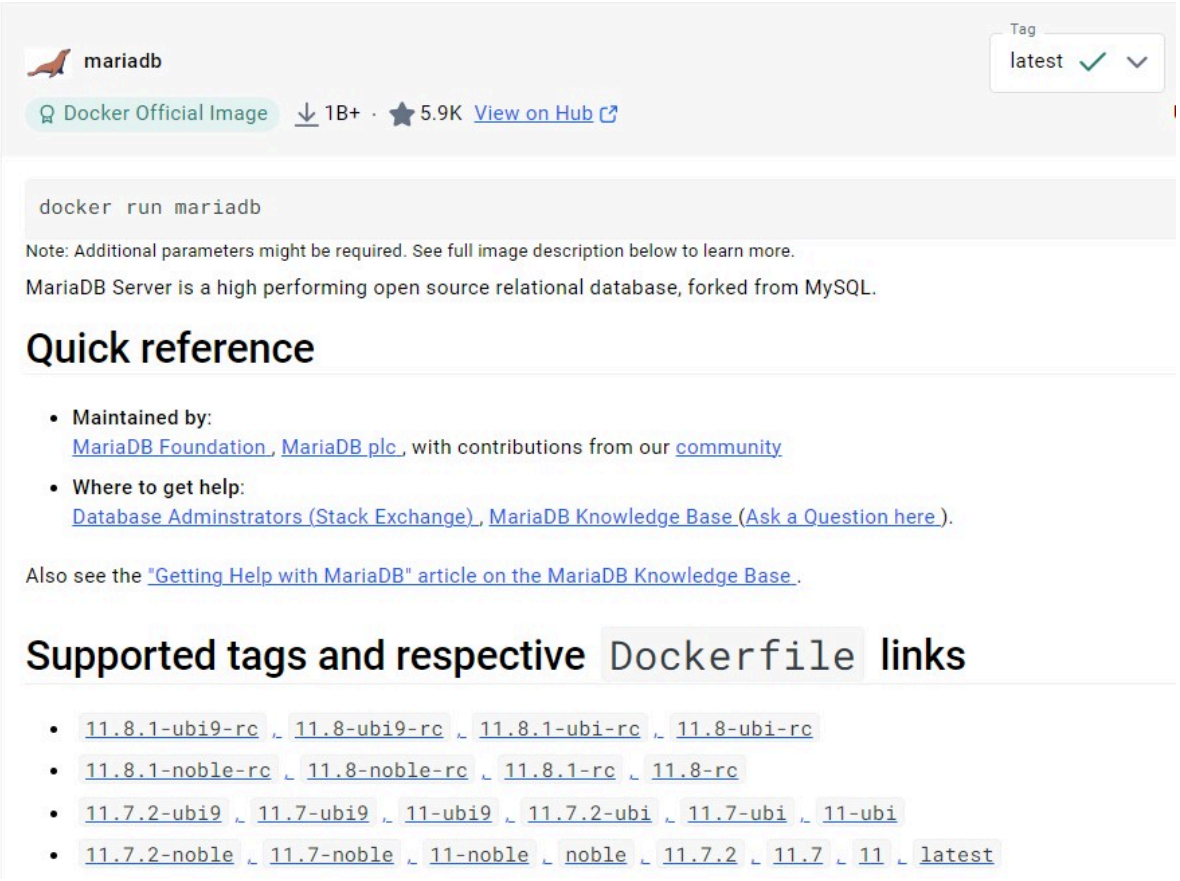

### 1.3 Creación contenedor con la imagen `mariaDB`

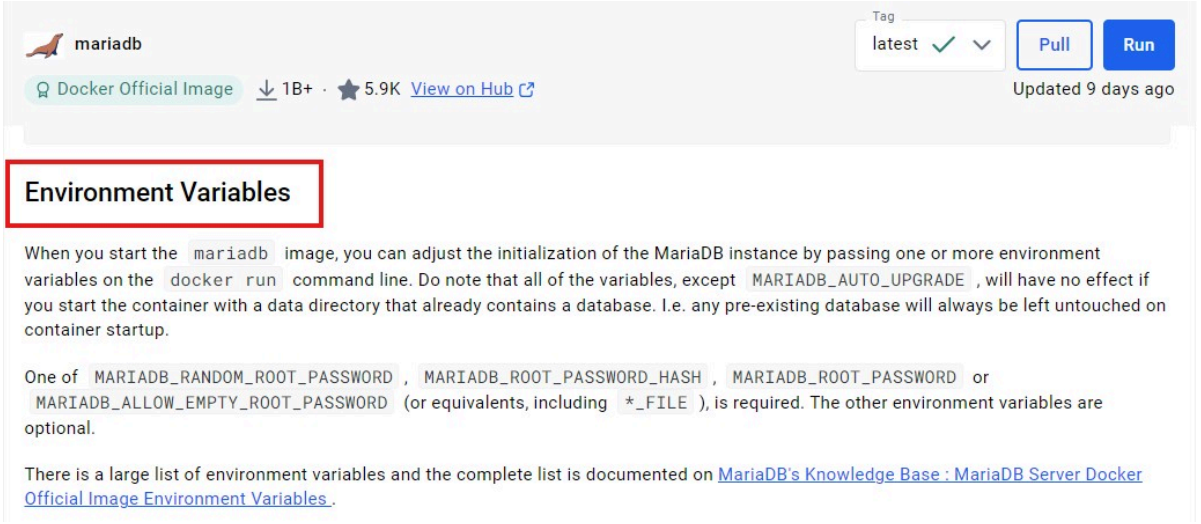
- En primer lugar buscaremos en el navegador de `Docker Desktop` la imagen `mariaDB`:



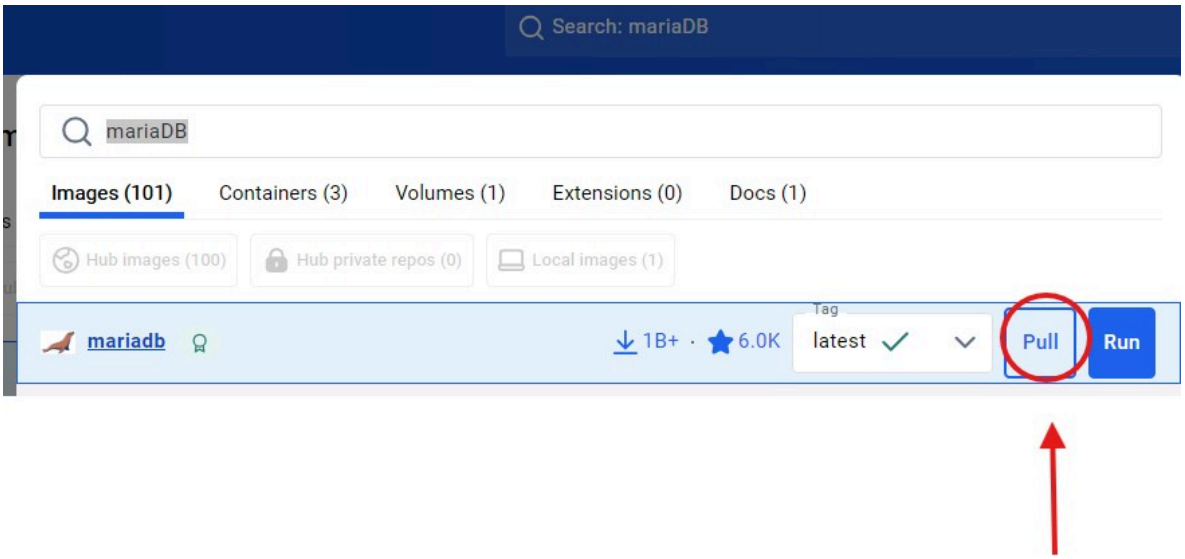

- A continuación, examinaremos nuestra imagen. Este apartado nos mostrará información relativa a la imagen: uso, cómo crear un contenedor a partir de ella, las versiones de la misma, como crear un fichero `Dockerfile`. Nosotros vamos a prestar especial atención a la sección `Environment Variables`, para configurar las variables de entorno con las que definiremos la contraseña y el usuario de nuestra base de datos:



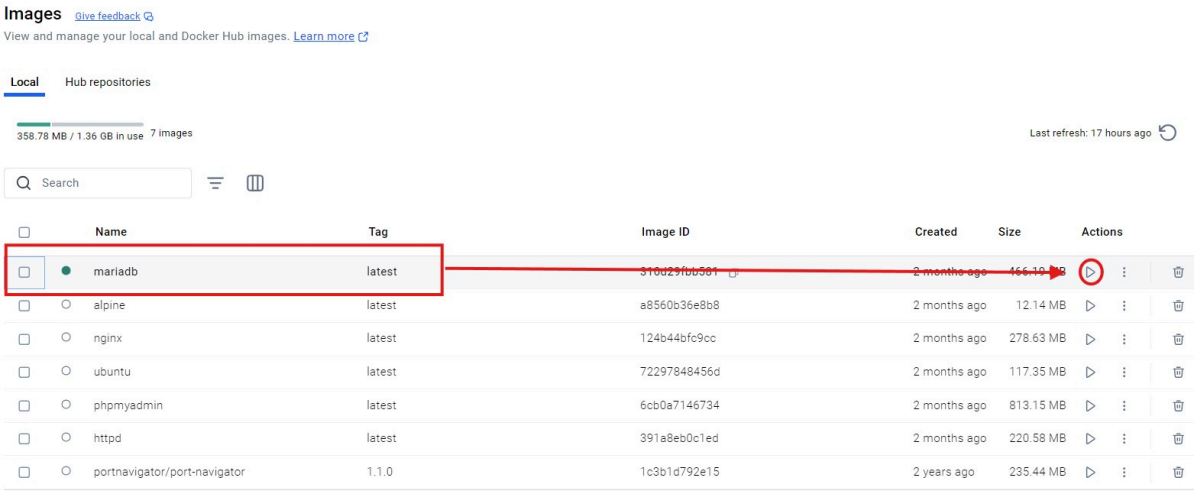



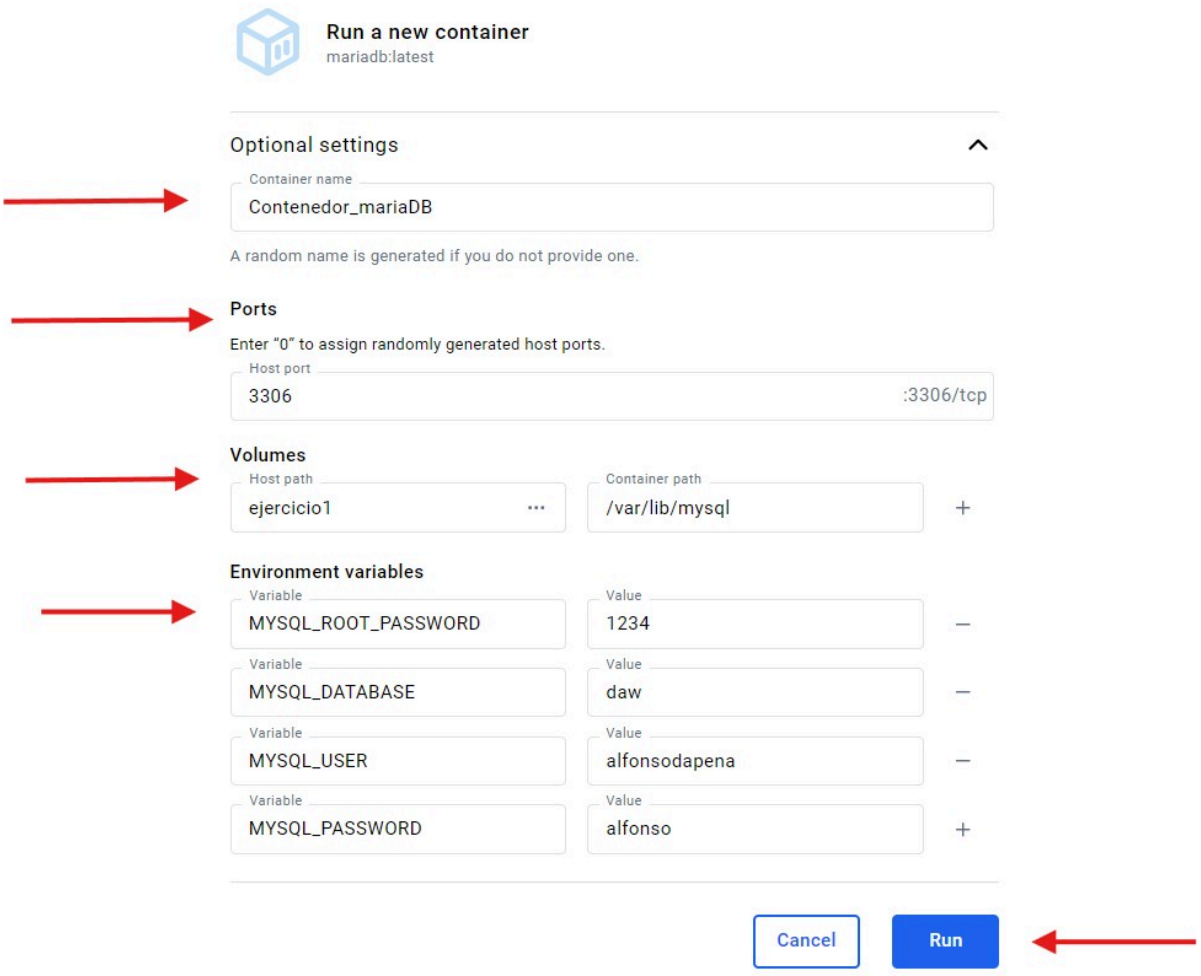

- A continuación, descargamos la imagen con la opción `pull`:

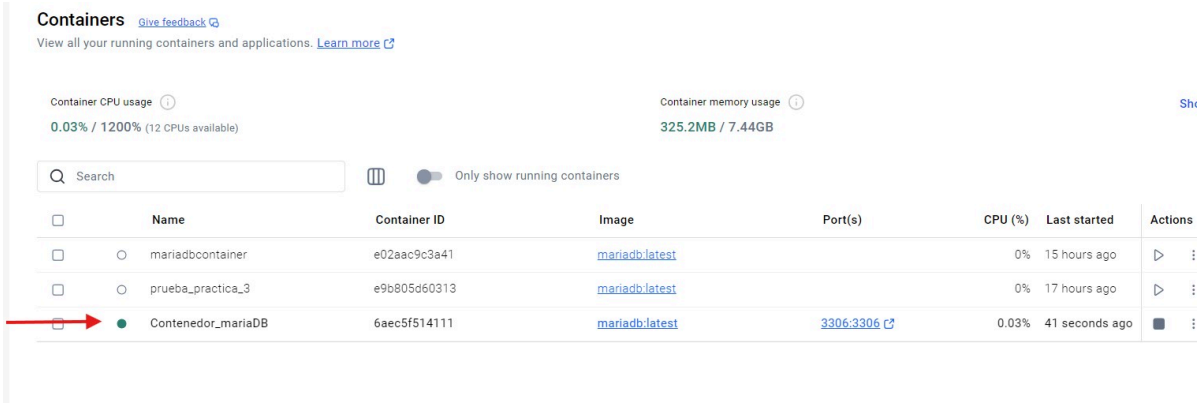



- Una vez hemos descargado nuestra imagen, procedemos a crear el contenedor desde la sección de imágenes y pinchando en el botón de `play`, situado debajo de `Actions`:



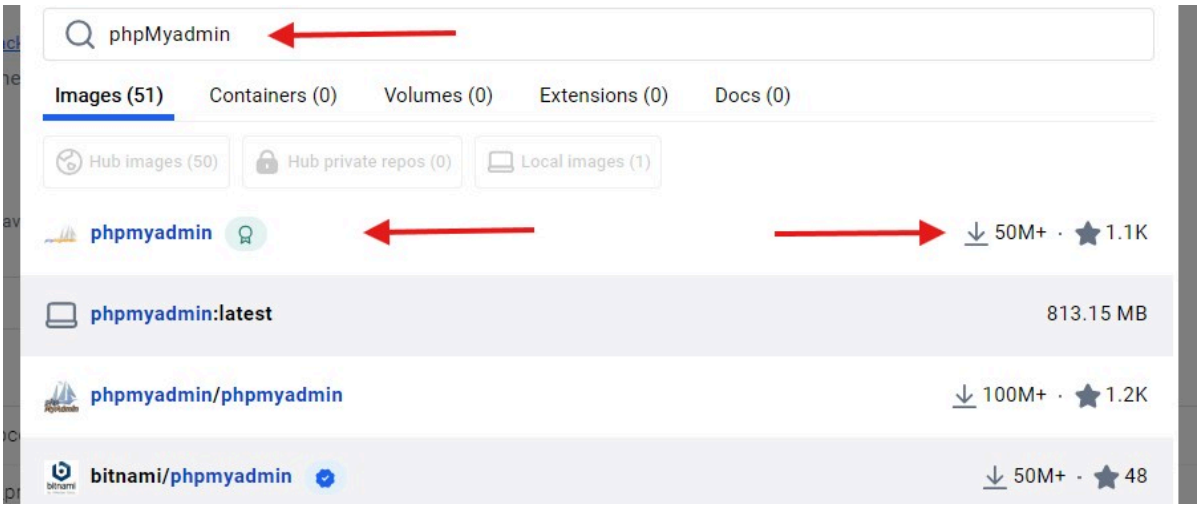

- A continuación, se nos abrirá una pestaña donde indicaremos el nombre de nuestro contenedor `mariaDB`, el puerto 3306 a través del que nuestro contenedor será accesible. En la opción `Volumes` crearemos el volumen `ejercicio1` en el directorio `/var/lib/mysqul`, para que el almacenamiento de nuestra base de datos sea persistente. En la pestaña `Environment variables`, introduciremos nuestra contraseña root `MYSQUL_ROOT_PASSWORD`, 1234, el nombre de la base de datos por defecto `daw`, un usuario `MYSQL_USER`, "alfonsodapena", con su contraseña `MYSQUL_PASSWORD`, " alfonso". Después de introducir todos los datos, pulsamos `RUN` para crear nuestro contenedor, que se inicializará de inmediato.

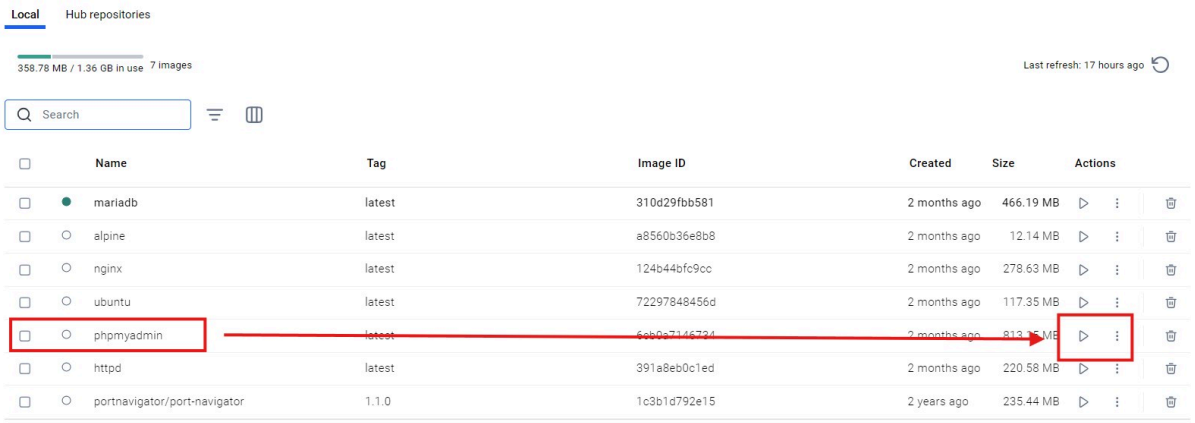


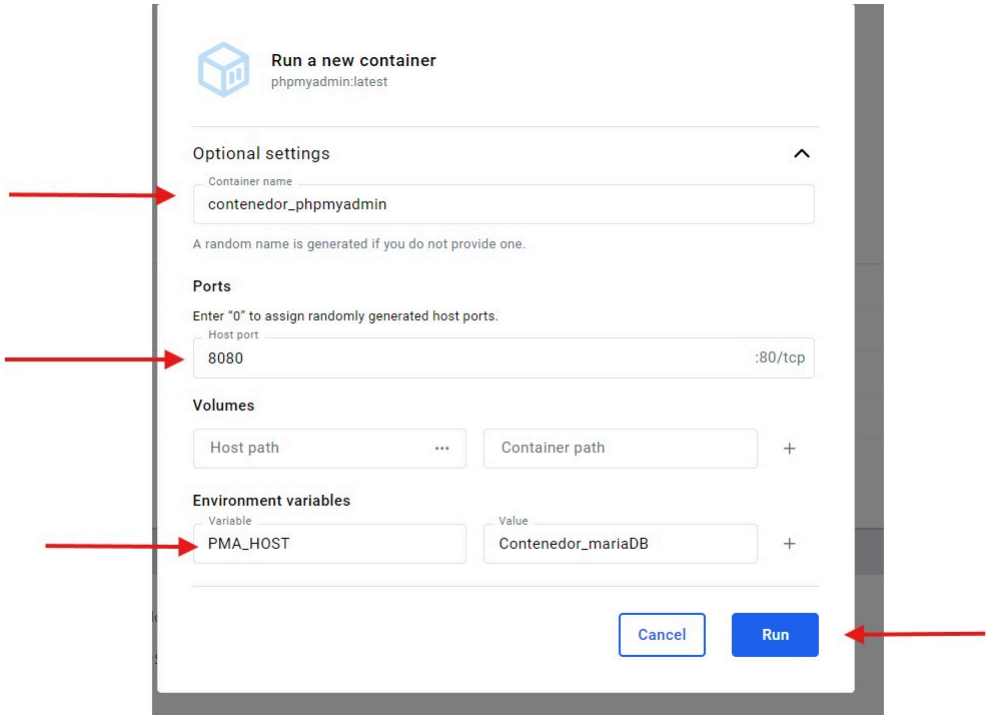




### 1.4 Creación contenedor `phpMyAdmin`.

- Para la creación de nuestro contenedor `contenedor_phpmyadmin`, seguiremos los pasos indicados en la creación del contenedor `Contenedor_mariaDB`, del punto 1.2:
    1. Buscamos la imagen en el navegador.
    2. Examinamos la imagen.
    3. Bajamos la imagen y creamos el contendor, con sus variables de entorno, donde indicaremos la variable de entorno `PMA_HOST` , que especifica el `host MySQL` para `phpMyAdmin`.



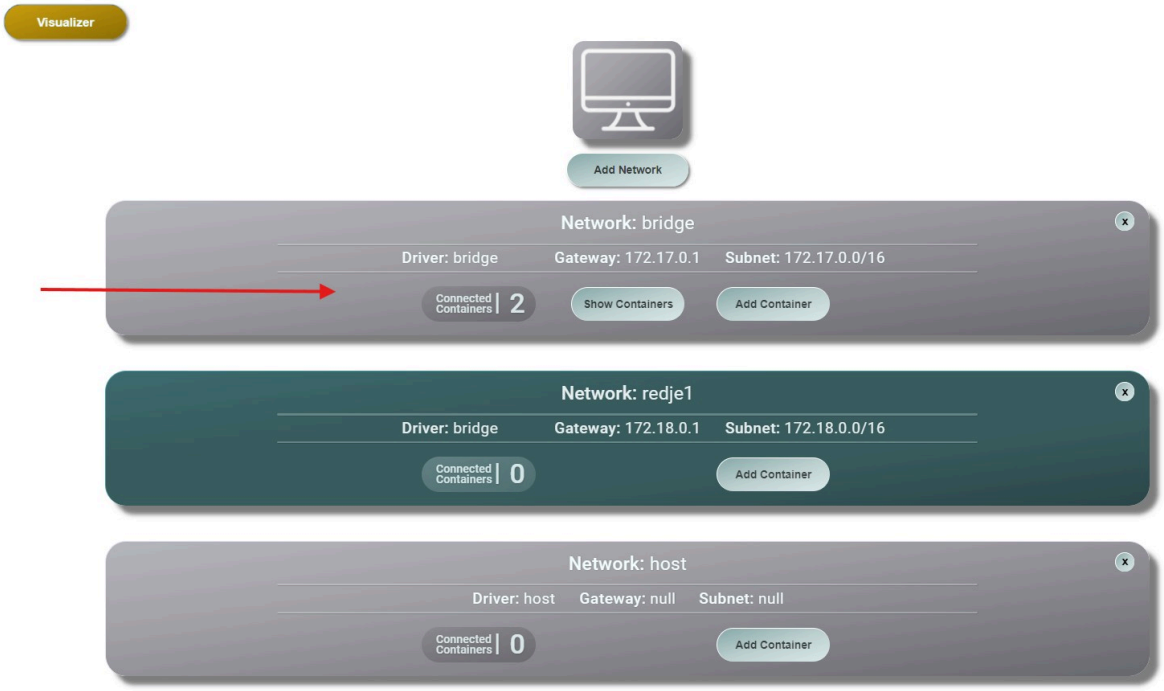



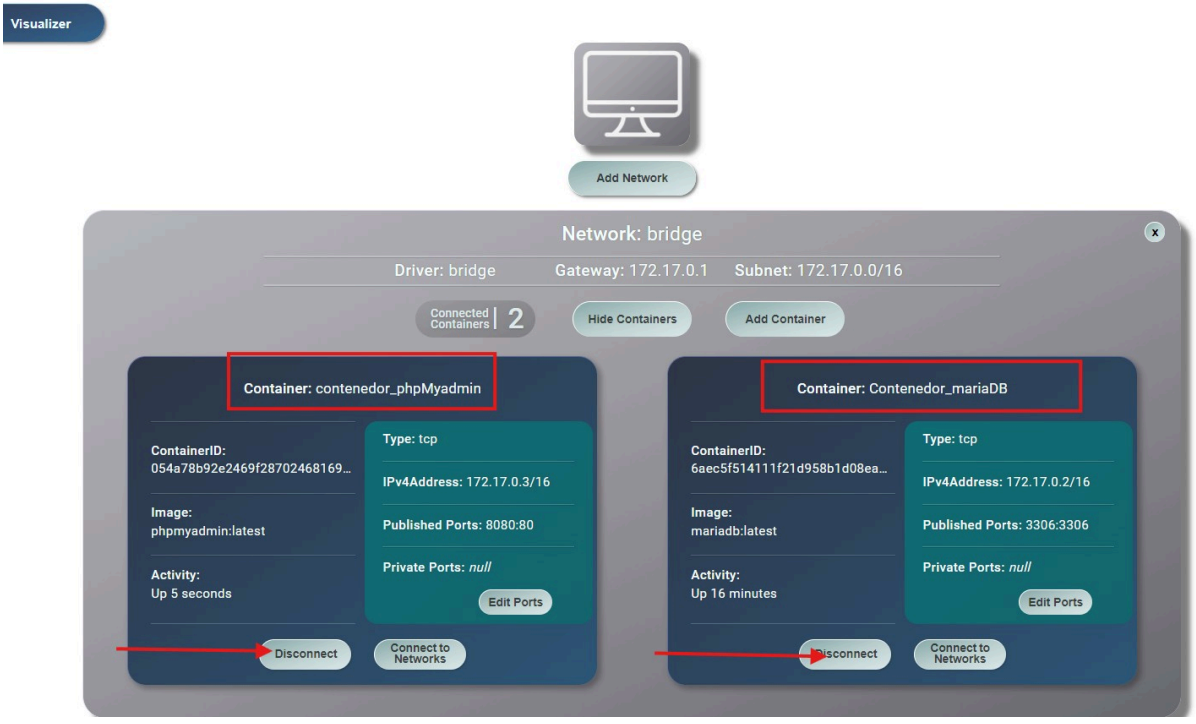

### 1.5 Conexión de contenedores a la red `redje1`.

- El siguiente paso será conectar a los dos contenedores a la misma red, en este caso la `redje1`. Por defecto `Docker Desktop`, al crear los contenedores los asigna a un red, por lo que antes deberemos de desconectarlos de la red a la que han sido asignados los contenedores automáticamente:

1.Conexión por defecto:

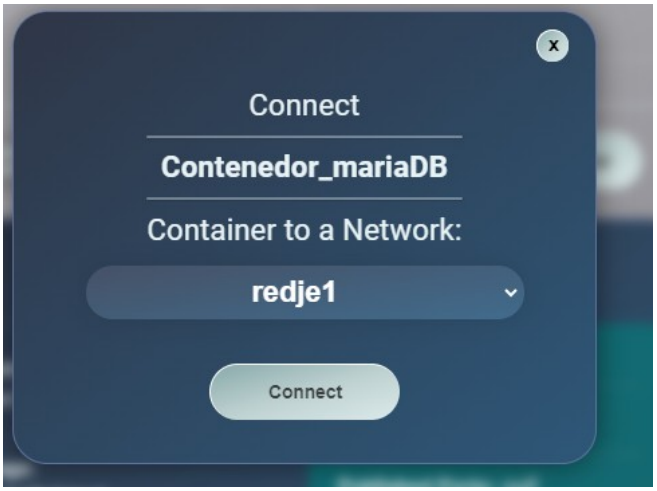


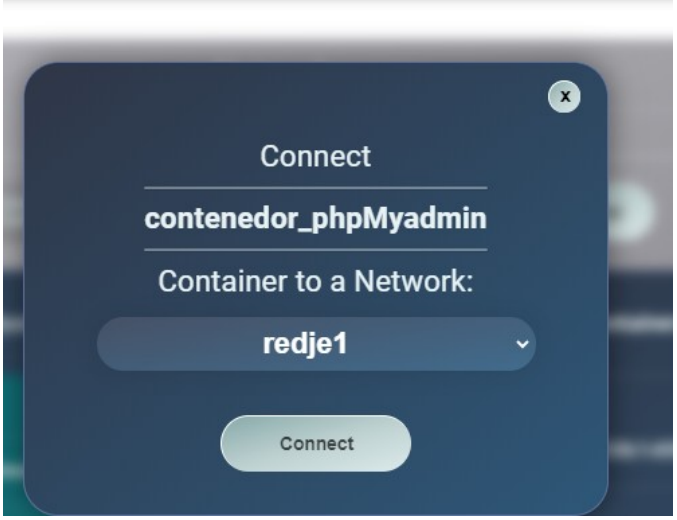

2. Procedemos a desconectarlos de la red

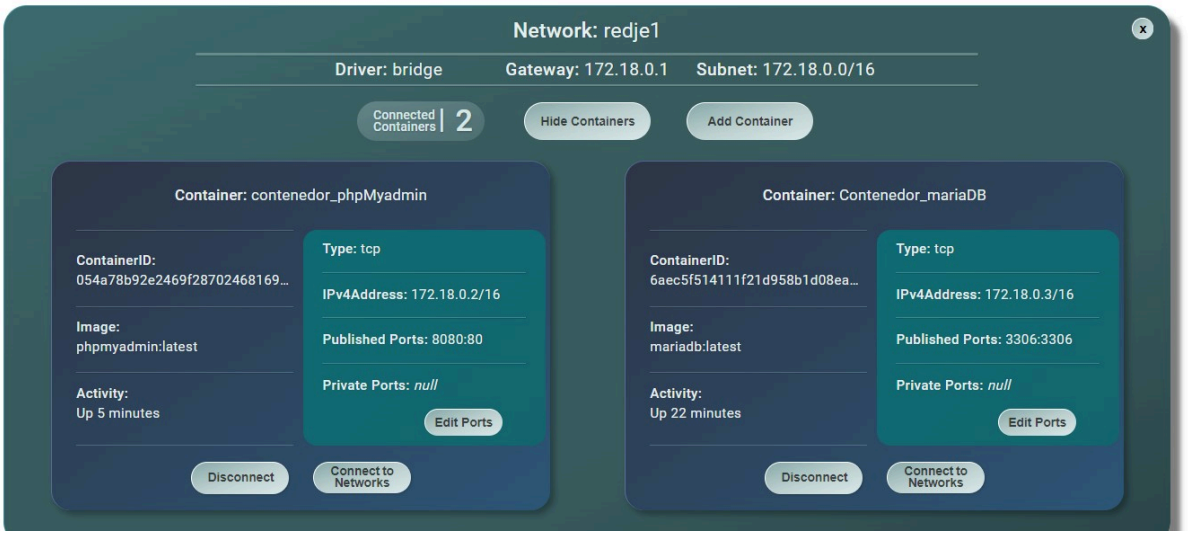



3. Después de desconectarlos, nos fijamos en que en la parte de abajo, figuran ya los dos contenedores desconectados por lo que ya podemos proceder a conectarlos a la red `redje1`


Driver: bridge
Gateway: 172.18.0.1
Subnet: 172.18.0.0/16
Connected Containers 0
Add Container
Network: host
Driver: host
Gateway: null
Subnet: null
Connected Containers 0
Add Container
Network: none
Driver: null
Gateway: null
Subnet: null
Connected Containers 2
Show Containers
Add Container
Network: none
Driver: null
Gateway: null
Subnet: null
Connected Containers 2
Hide Containers
Add Container
Container: contenedor_phpMyadmin
ContainerID:
054a78b92e2469f28702468169...
Image:
phpmyadmin:latest
Activity:
Up 2 minutes
Type:
IPv4Address:
Published Ports: null
Private Ports: null
Edit Ports
Connect to Networks
Container: Contenedor_mariaDB
ContainerID:
6aec5f514111f21d958b1d08ea...
Image:
mariadb:latest
Activity:
Up 19 minutes
Type:
IPv4Address:
Published Ports: null
Private Ports: null
Edit Ports
Connect to Networks


Connect
Contenedor_mariaDB
Container to a Network:
redje1
Connect

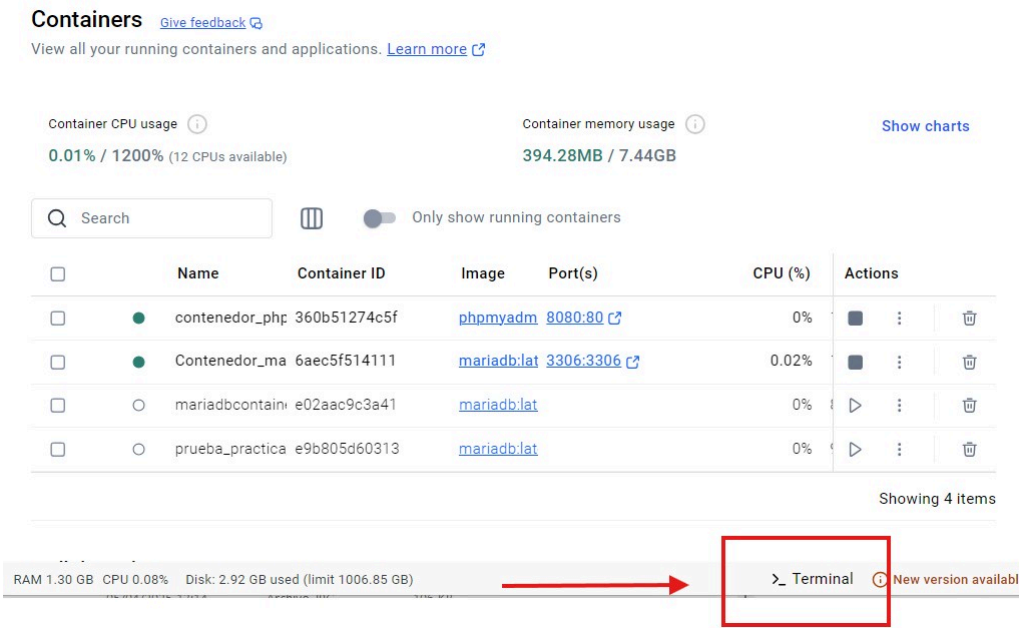

Connect
contenedor_phpMyadmin
Container to a Network:
redje1
Connect

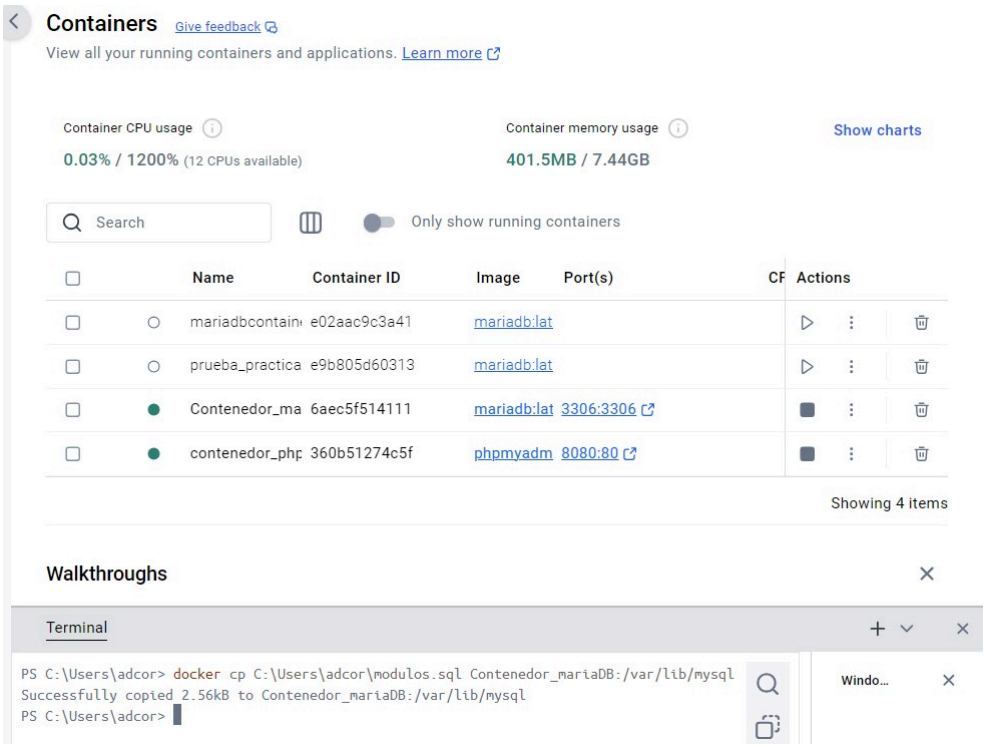



### 1.6 Creación script `modulos`, y ejecución del mismo en nuestra base de datos `DAW`

- En primer lugar debemos de crear un script SQUL, que cree una tabla `modulos`. Una vez creado este archivo, copiaremos el mismo al fichero raíz del `Contenedor_mariaDB` creado. Para ello abrimos la terminal integrada de `Docker Desktop`, y copiamos nuestro archivo `modulos.sql` al directorio `/var/lib/mysql`:

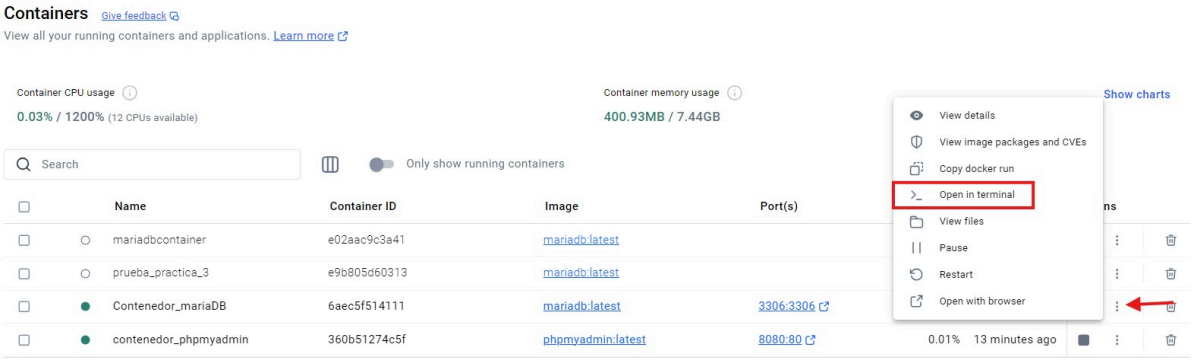
```
$ docker cp C:\Users\adcor\modulos.sql Contenedor_mariaDB:/var/lib/mysql
```

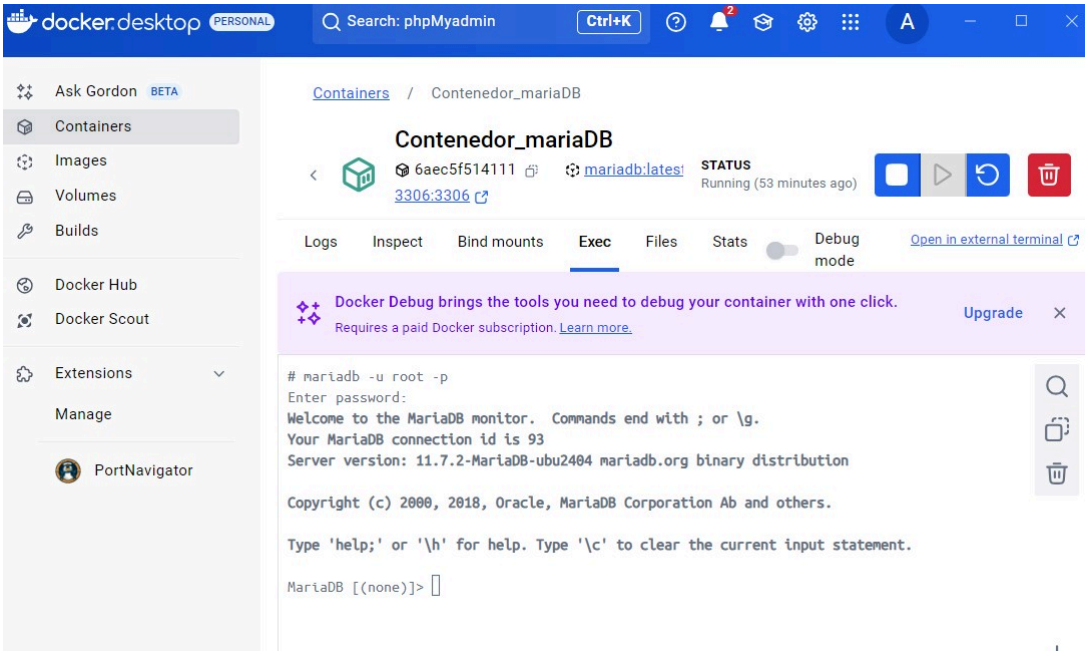





- A continuación, accedo al `Contenedor_mariaDB`, y en la pestaña `Exec`, accedo al contenedor y ejecuto el siguiente comando para acceder a al base de datos

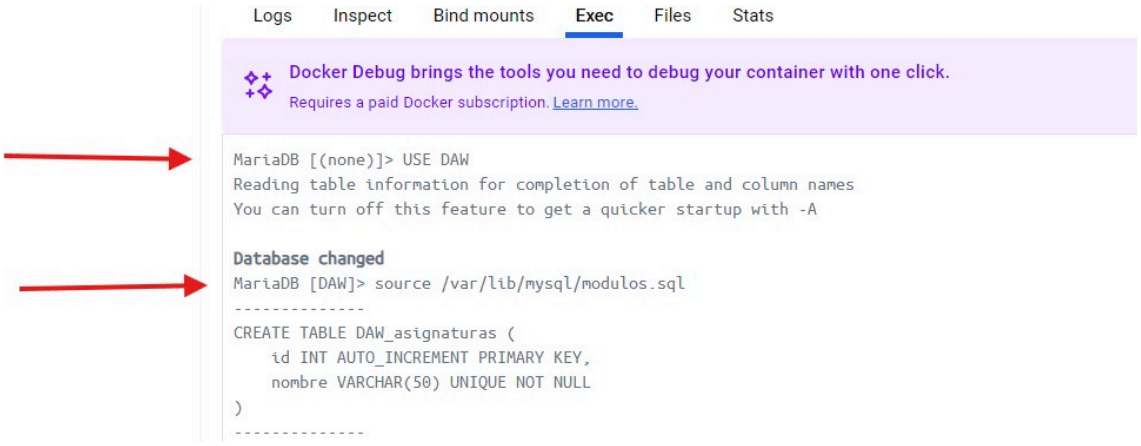
```
mariadb -u root -p
```



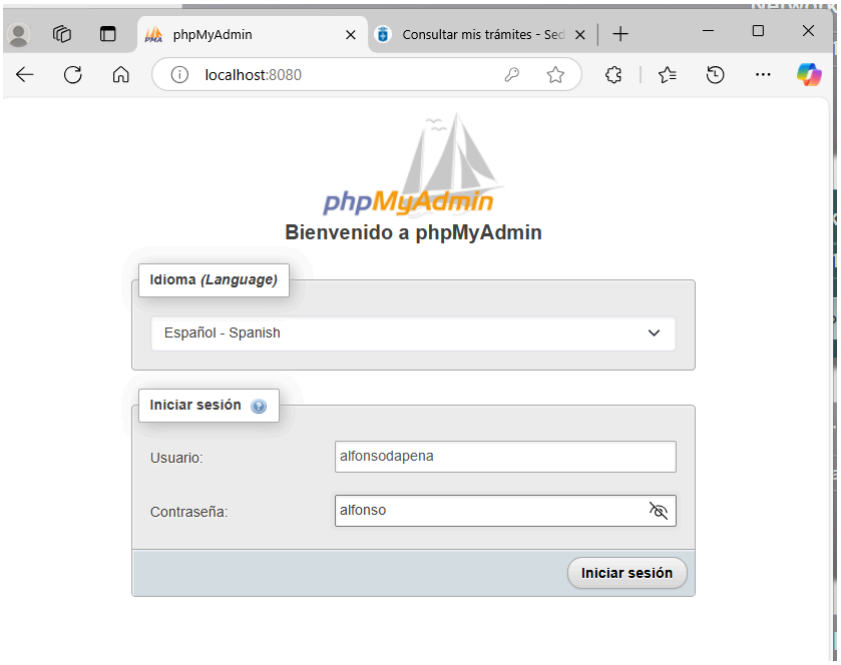
- A continuación, cambiamos a la base de datos `DAW` y ejecuto el script `modulos.sql`, para crear la tabla `modulos`. Para ello introducimos las siguientes instrucciones:

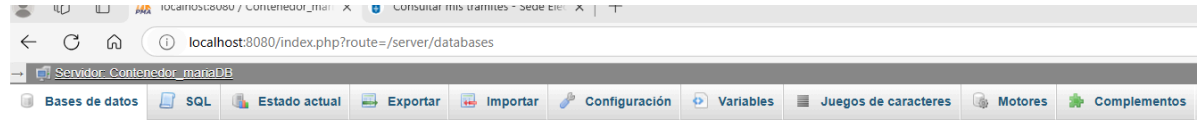
```
USE DAW
source /var/lib/mysql/moduls.squl
```




### 1.7 Acceso a la base de datos.

- Para comprobar que nuestra base de datos funciona correctamente, accedemos a nuestro navegador indicando `localhost:8080`, para acceder a la aplicación `phpMyAdmin`, que nos mostrará la tabla `modulos`.






## Bases de datos

Crear base de datos

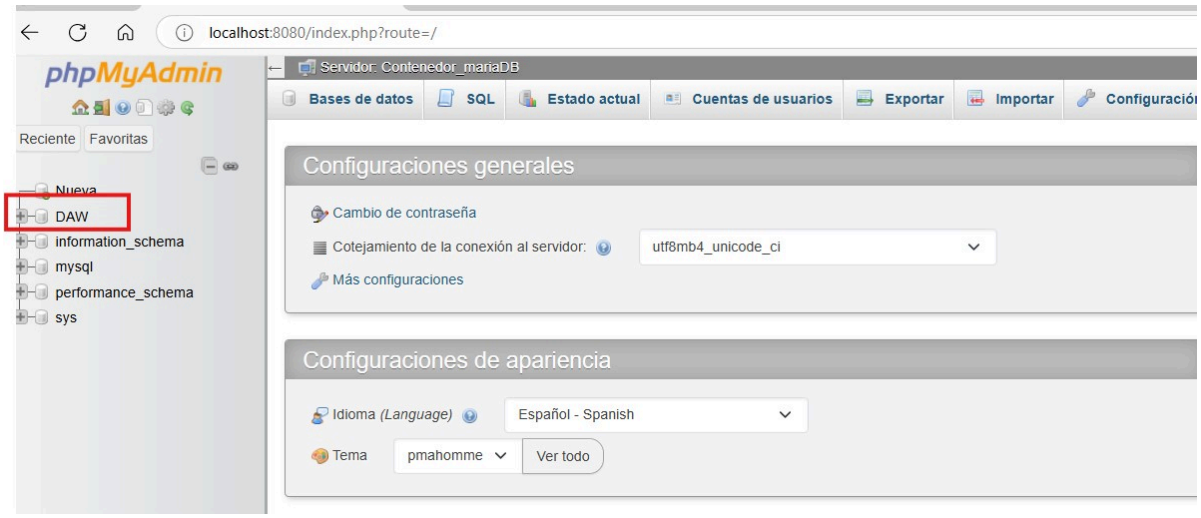
No tienes permisos para crear bases de datos

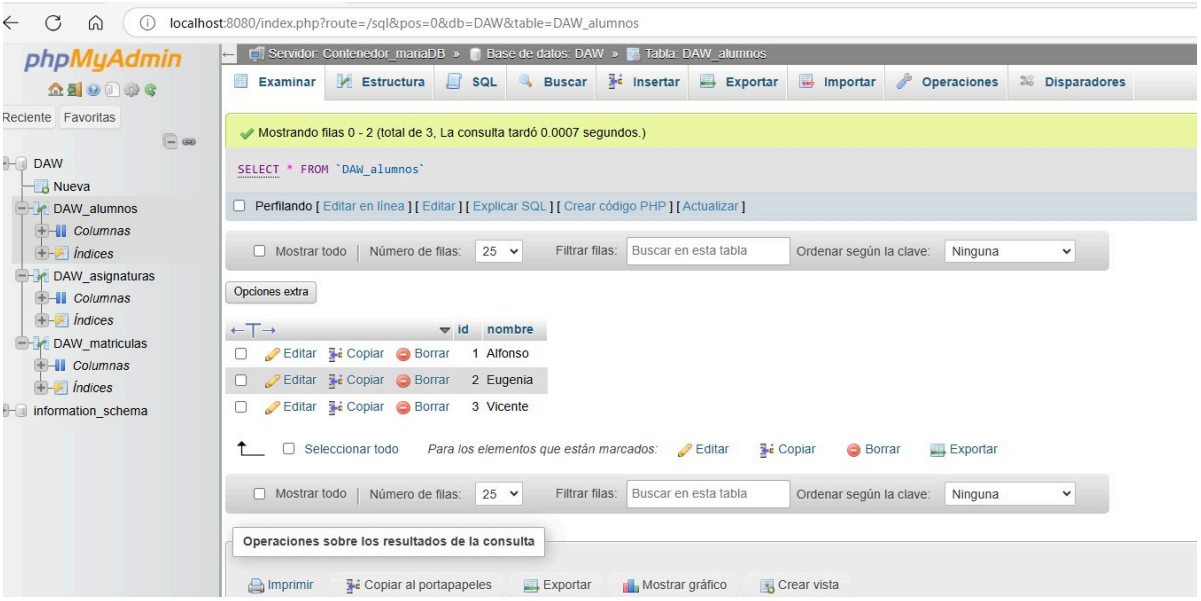
| Base de datos | Cotejamiento | Acción |
|---|---|---|
| DAW | utf8mb4_uca1400_ai_ci | Seleccionar privilegios |
| information_schema | utf8mb3_general_ci | Seleccionar privilegios |

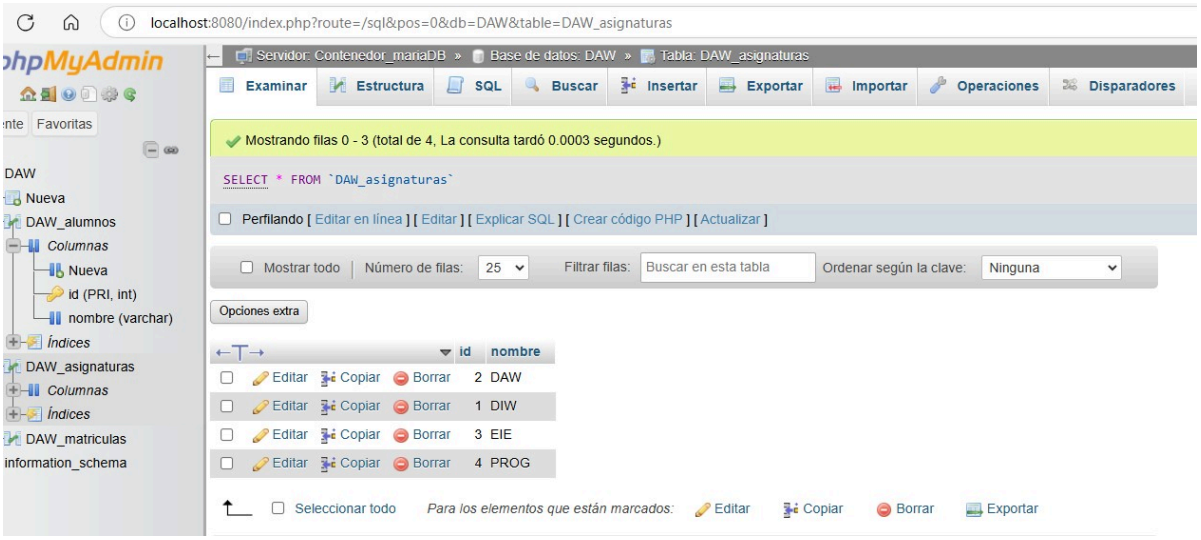
**Total: 2**

Nota: Activar aquí las estadísticas de la base de datos podría causar tráfico pesado entre el servidor web y el servidor MySQL.

Activar estadísticas

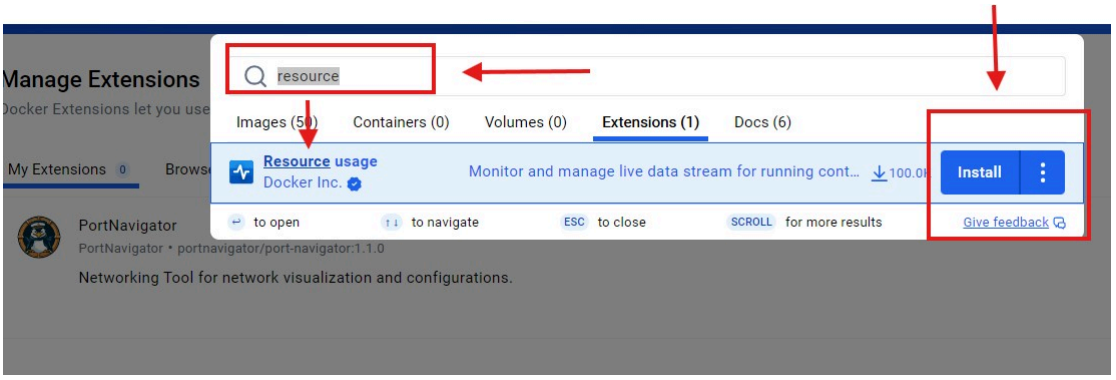


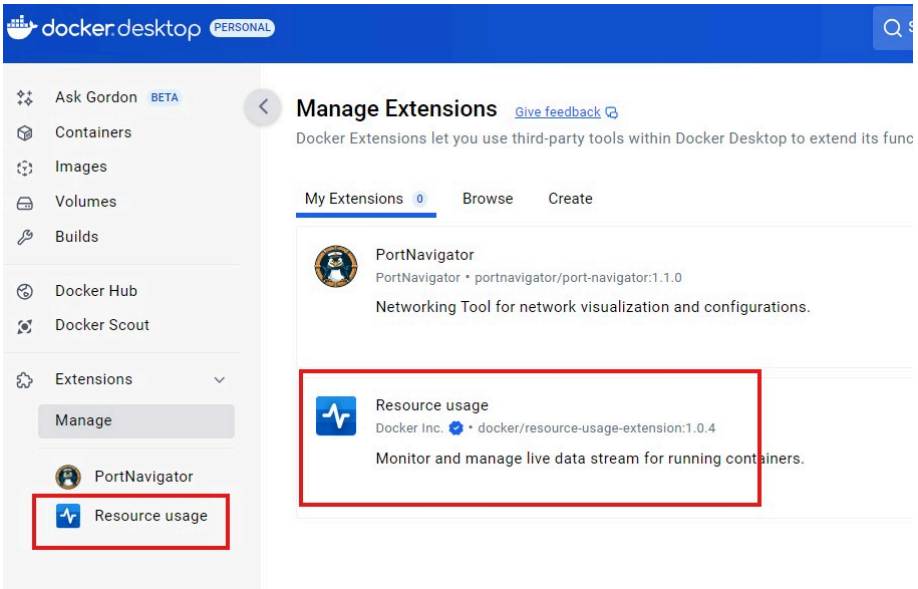





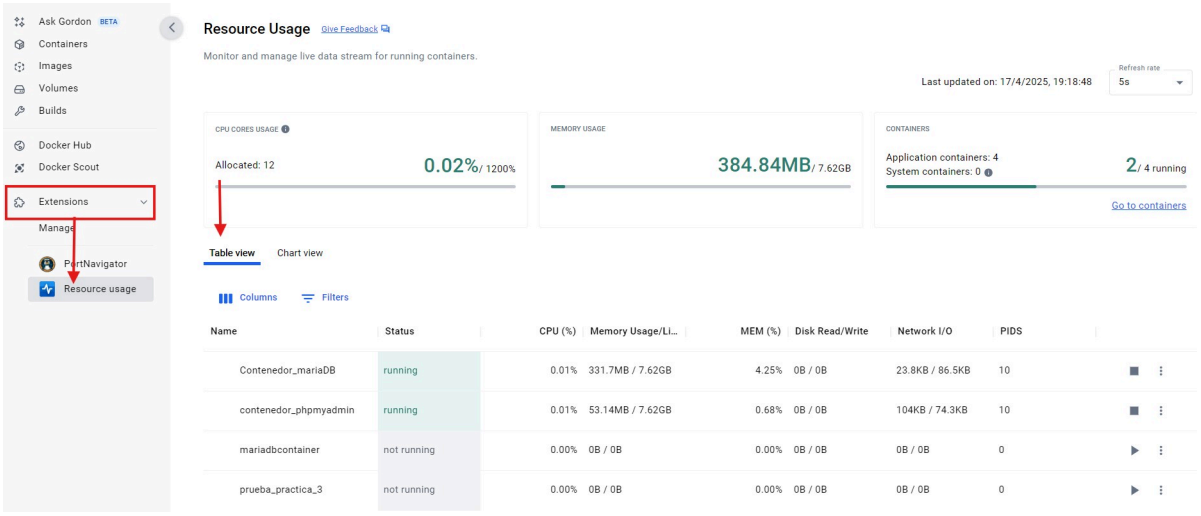

### 1.8 Instalamos extensión `Resource usage`

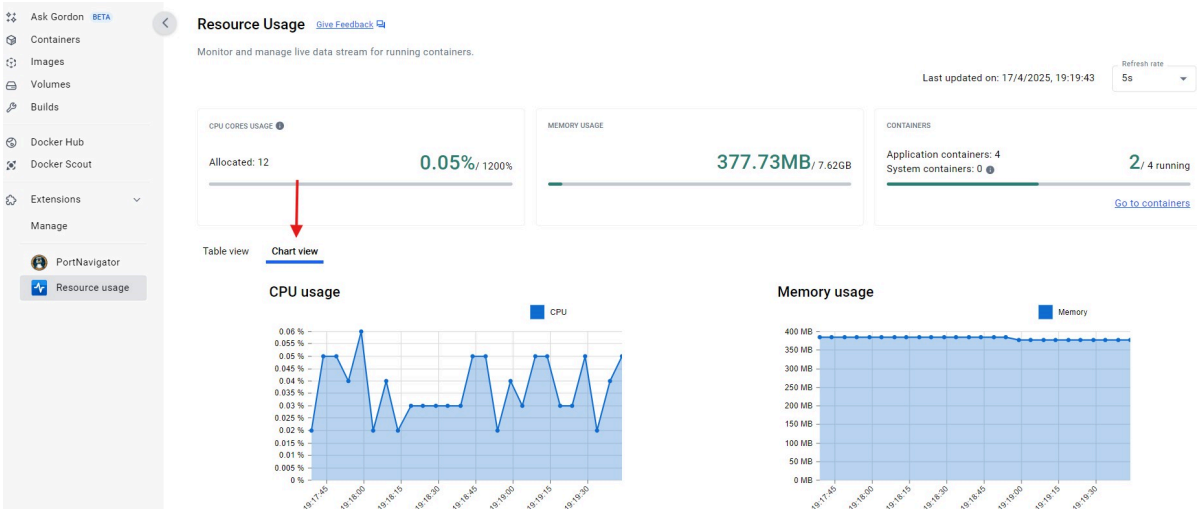
- Del mismo modo que instalamos la extensión `PortNavigator`, accedemos a la pestaña de búsqueda en la aplicación, realizamos su búsqueda y la instalamos:




- Una vez instalada, arrancamos nuestros contenedores y comprobamos nuestra salida cuando estos están activos. La extensión permite dos vistas: `Table view`, y `Chart view`.


Ask Gordon BETA
Containers
Images
Volumes
Builds
Docker Hub
Docker Scout
Extensions
Manage
PortNavigator
Resource usage
Resource Usage
Give Feedback
Monitor and manage live data stream for running containers.
Last updated on: 17/4/2025, 19:18:48
Refresh rate
5s
CPU CORES USAGE
Allocated: 12
0.02% / 1200%
MEMORY USAGE
384.84MB / 7.62GB
CONTAINERS
Application containers: 4
System containers: 0
2 / 4 running
Go to containers
Table view
Chart view
Columns
Filters
Name
Status
CPU (%)
Memory Usage/Li...
MEM (%)
Disk Read/Write
Network I/O
PIDS
Contenedor_mariaDB
running
0.01%
331.7MB / 7.62GB
4.25%
0B / 0B
23.8KB / 86.5KB
10
contenedor_phpmyadmin
running
0.01%
53.14MB / 7.62GB
0.68%
0B / 0B
104KB / 74.3KB
10
mariadbcontainer
not running
0.00%
0B / 0B
0.00%
0B / 0B
0B / 0B
0
prueba_practica_3
not running
0.00%
0B / 0B
0.00%
0B / 0B
0B / 0B
0


Ask Gordon BETA
Containers
Images
Volumes
Builds
Docker Hub
Docker Scout
Extensions
Manage
PortNavigator
Resource usage
Resource Usage
Give Feedback
Monitor and manage live data stream for running containers.
Last updated on: 17/4/2025, 19:19:43
Refresh rate
5s
CPU CORES USAGE
Allocated: 12
0.05% / 1200%
MEMORY USAGE
377.73MB / 7.62GB
CONTAINERS
Application containers: 4
System containers: 0
2 / 4 running
Go to containers
Table view
Chart view
CPU usage
CPU
0.06 %
0.055 %
0.05 %
0.045 %
0.04 %
0.035 %
0.03 %
0.025 %
0.02 %
0.015 %
0.01 %
0.005 %
0 %
Memory usage
Memory
400 MB
350 MB
300 MB
250 MB
200 MB
150 MB
100 MB
50 MB
0 MB